An Efficient Lossless Compression Using Double Huffman Minimum Variance Encoding Technique

Sandeep G. S[1], Sunil Kumar B. S[2], Deepak D J[3]
Department of Computer Science and Engineering[1, 2, 3]
GM Institute of Technology, Davangere, Karnataka, India
sandeepgsgagan@gmail.com[1] sunilkumarbs2007@gmail.com[2] deepakdoggalli@gmail.com[3]

Abstract: *A Huffman code is a particular type of optimal prefix code that is commonly used for loss-less data compression. The process of finding such a code is known as Huffman coding. The output from Huffman's algorithm can be viewed as a variable-length code table for encoding a source symbol. The algorithm derives this table from the estimated probability or frequency of occurrence for each possible value of the source symbol. In this paper, we present a new approach to measure the performance and redundancy that work on two methods of coding like Huffman coding and Minimum Variance Huffman Coding. After getting the code-word for each symbol, we compress it on the basis of its binary values like 0 and 1 using binary coding. This is applied to both the approaches; this process is called as Double Huffman Coding. Finally we produce a better result than Huffman coding.*

Keywords*: Average length, Data Compression, Double Huffman coding, Entropy, Minimum Variance.*

I. INTRODUCTION

The term "Data Compression" means compression of data as much as possible from its original size. But in this communication, we always want to transmit data efficiently and noise free. We try to find such data that are loss-less. Lossless compression refers to compression methods for which the original uncompressed data-set can be recovered exactly from the compressed stream. In computer science, lossless compression is very important for the compression of text, digitized medical data, etc. The Huffman encoding algorithm provides an optimal way to compress dataset without any loss of information [3]. The semi static Huffman compression involves two passes over a dataset during encoding: one to gather symbol frequency information and the other to perform actual coding after a code has been constructed [4]. Huffman coding uses a specific method for choosing the representation for each symbol, resulting in a prefix code [1]. Huffman coding is such a widespread method for creating prefix codes that the term "Huffman code" is widely used as a synonym for "prefix code" even when such a code is not produced by Huffman's algorithm. The Huffman coding gives more emphasize to those symbols in a dataset which has high probability of occurrence in comparison with those with lower probability of occurrences [5]. Although this technique gives a good compression for those dataset in which the frequency of occurrence of symbols are high, but for lower frequency data Huffman coding encodes with relatively higher number of bits. Instead of using this straight forward Huffman coding, in this paper we use Huffman coding in a novel way, we create a string of the symbol after concatenation in the alphabetic or numeric order, and compress that using 0 and 1 probability.

II. PROPOSED METHOD

The process begins with the leaf nodes containing the probabilities of the symbol they represent, and then a new node whose children are the two nodes with smallest probability is created, such that the new node probability is equal to the sum of its children's probability. With the previous two nodes merged into one node and with the new node is considered. The procedure is repeated until only one node remains in the Huffman tree. Huffman coding uses a specific method for choosing the representation for each symbol, resulting in a prefix code that expresses the most common source symbols using shorter strings of bits than are used for less common source symbols. Huffman was able to design the most efficient compression method of this type.

STEPS OF DOUBLE HUFFMAN CODING

Huffman algorithm works from leaves to the root in the opposite direction. The algorithm has following steps,

1. Initially all the probabilities are arranged in Ascending order.
2. Create a leaf node for each symbol and add it to frequency of occurrence.
3. While there is more than one node in the queue:
   i. Remove the two nodes of lowest probability or frequency from the queue.
   ii. Assign 0 to the second smallest element and 1 to the first smallest element.
   iii. Create a new internal node with these two nodes as children and with probability equal to the sum of the two nodes' probabilities.
   iv. Add the new node to the queue.
4. The remaining node is the root node and the tree is complete.
5. Compute the Entropy, Average Length and Redundancy (Round-2 Ends).
6. After Getting the Codeword of each Symbol/Alphabet, concatenate it.
7. Find the Probability of 0 and 1.
8. Repeat Step form 1-5.

CALCULATION OF ENTROPY, AVERAGE LENGTH AND REDUNDANCY

Entropy**:** It is the smallest number of bits needed, on an average, to represent a symbol. Shannon borrowed the definition of *entropy [6]* from statistical physics to capture the notion of how much information is contained in a message and their probabilities. For a set of possible messages S,
Shannon defined entropy as,

$$H(S) = -\sum_{x \in s} p(s) \log_2 p(s) \quad (1)$$

or

$$Entropy = -\sum_{i=1}^{6} P(x_i) \log_2 P(x_i) \quad (2)$$

where p(s) is the probability of message s.

Average Length: Given a probability distribution on a set of messages and associated variable length code C, we define the *average length* of the code as

$$l_a(C) = \sum_{(s,w) \in C} p(s) * l(w) \quad (3)$$

where, l(w) is the length of the codeword w. We say that a prefix code C is an *optimal* prefix code if l_a (C) is minimized.

Redundancy: The Difference between Entropy and average length, and is given by the formula,

Redundancy = Average Length - Entropy (4)

III. IMPLEMENTATION

Building of Huffman code can be done with Huffman coding algorithm and Minimum Variance Huffman coding algorithm. There are two different approaches presented in this section. The first one is Huffman with Binary coding, as it was presented in the section - A . However we have proposed a second approach where in the first encoding is by Minimum Variance Huffman Coding[2] and then by Binary encoding method in section - B, which yields a better efficiency. In order to give a comparison of the two methods, both approaches are illustrated here under.

APPROACH-A:- Huffman-Binary Coding

This method has the following steps to carryout

- Calculation of Entropy,
- Average Length and
- Redundancy calculation using Huffman coding for different samples.

These steps have been explained with an help of following example.

Example:

TABLE I: INPUT ALPHABETS AND ITS ASSOCIATIVE PROBABILITY.

Alphabet	A	B	C	D	E	F
Probability	0.30	0.30	0.13	0.12	0.10	0.05

First Round: Huffman Coding

Steps 1-4: as shown using below tree construction in Figure-1.

After Apply the algorithm Codeword for the Alphabets are:-
A = 00, B = 01, C = 100, D = 101, E = 110, and F = 111

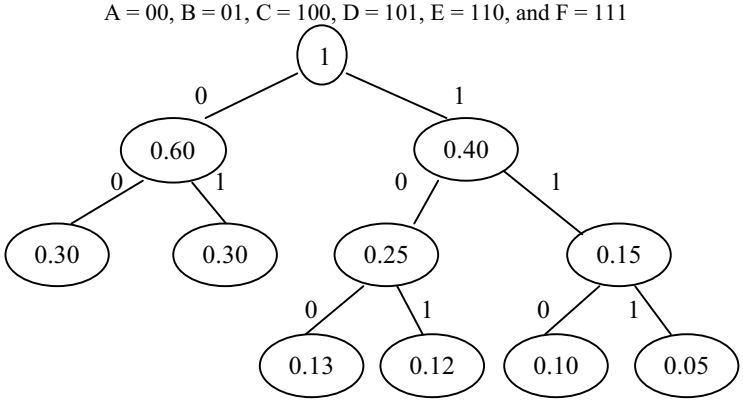

Fig 1: Huffman Tree

Step 5: Entropy, Average length, Redundancy calculations are done here

Entropy =-0.30$\log_2$0.30 - 0.30$\log_2$0.30 - 0.13$\log_2$ 0.13 - 0.12 $\log_2$ 0.12 - 0.10 $\log_2$ 0.10 - 0.05 $\log_2$ 0.05
= 2.3375 bits / symbol.

Average Length =0.30*2+0.30*2+0.13*3+0.12*3
+ 0.10*3 + 0.05*3
= 2.4 bits/ symbol.

Redundancy = 2.4 - 2.3375
= 0.0625 bits/symbol

TABLE II: HUFFMAN CODING RESULTS

Probability	**Code word**	**Code word Length**	**Entropy (bits/ symbol)**	**Average Length (bits/ symbol)**	**Redundancy (bits/ symbol)**
0.30	00	2	2.3375	2.4	0.0625
0.30	01	2			
0.13	100	3			
0.12	101	3			
0.10	110	3			
0.05	111	3			

Step-6: Concatenate the alphabet with codeword, so the codeword string is "0001100101110111".
This String will contain only two binary numbers 0 and 1.

Step-7: The Probability for these numbers is

P (0) =7/16.
P (1) =9/16.

Repeat the Procedure from Step 1-5 discussed in section 2.1. The steps 1-4 are shown Figure 2.

Second Round: Binary Encoding

Steps 1-4 are represented using Tree construction and are shown below:

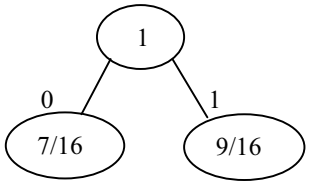

Fig 2: Binary Huffman Tree

Step 5:

Entropy = 7/16 $\log_2$ 7/16 + 9/16 $\log_2$ 9/16
= 0.9871 bits/symbol.

Average Length = 7/16 + 9/16
= 1 bits / symbol.

Redundancy = 1 – 0.987
= 0.012 bits / symbol.

After Applying Proposed Double Huffman Coding, the following results are obtained.

TABLE III: DOUBLE HUFFMAN CODING RESULTS

Probability	Code word	Code word Length	Entropy (bits/ symbol)	Average Length (bits/ symbol)	Redundancy (bits/ symbol)
P(0) = 9/16	0	1	0.9871	1	0.012
P(1) = 7/16	1	1			

APPROACH-B:- Minimum Variance Huffman coding-Binary Coding

Calculation of Entropy, Average Length and Redundancy using Min. Variance Huffman coding for samples shown in TABLE-III

First Round: Minimum Variance Huffman Coding
This method has following steps to perform.

Steps 1-4 as shown using below tree construction

After Apply the algorithm Codeword for the Alphabets are:-
A = 01, B = 00, C = 100, D = 101, E = 110, and F = 111

The Figure 3 shows Minimum variance Huffman tree

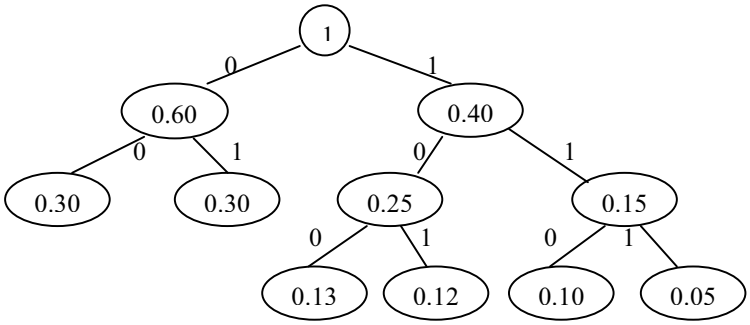

Fig 3: Minimum Variance Huffman Tree

Step 5:
Entropy = -0.30 $\log_2$0.30–0.30 $\log_2$0.30–0.13 $\log_2$0.13 – 0.12 $\log_2$ 0.12– 0.10 $\log_2$ 0.10 –0.05 $\log_2$ 0.05
=2.3375 bits / symbol.

Average Length = 0.30*2 + 0.30*2 + 0.13*3 + 0.12*3 + 0.10*3 + 0.05*3
= 2.4 bits / symbol.

Redundancy = 2.3375-2.4
= 0.0625 bits / symbol.

TABLE IV: MINIMUM VARIANCE HUFFMAN CODING RESULTS

Probability	Code word	Codeword Length	Entropy (bits/ symbol)	Average Length (bits/ symbol)	Redundancy (bits/ symbol)
0.30	00	2	2.3375	2.4	0.0625
0.30	01	2			
0.13	100	3			
0.12	101	3			
0.10	110	3			
0.05	111	3			

Step 6: Concatenate the Alphabet with Codeword, so the Codeword String is "0001100101110111" This String will contain only two binary numbers 0 and 1.

Step 7: So the Probability for these numbers is P (0) =7/16 and P (1) =9/16.

Repeat the Procedure from Step 1-5 discussed in section 2.1. The steps 1-4 are shown Figure 4.

Second Round : Binary Encoding
Steps 1-4 are represented using Tree construction and is shown below.

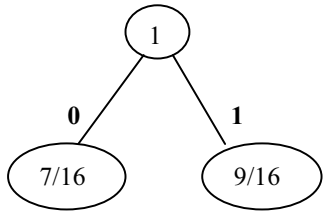

Fig 4: Binary Minimum Variance Huffman Tree
Step 5: Calculation of Entropy, Average Length and Redundancy

Entropy= 7/16 $\log_2$ 7/16 + 9/16 $\log_2$9/16
= 0.9871bits/symbol.
Average Length = 7/16 + 9/16
= 1 bits/symbol.
Redundancy = 1 – 0.987
= 0.012 bits/symbol.

After Applying Proposed Minimum Variance with Binary Coding

TABLE-V: MINIMUM VARIANCE WITH BINARY CODING

Probability	Code word	Code word Length	Entropy (bits/ symbol)	Average Length (bits/ symbol)	Redundancy (bits/ symbol)
P(0) = 9/16	0	1	0.9871	1	0.012
P(1) = 7/16	1	1			

Similarly, we calculated for different samples of inputs and are enlisted below.

1) Letters = {A, B, C, D, E}
Probability = {0.4, 0.2, 0.2, 0.1, 0.1}
2) Letters = {A, B, C, D, E}
Probability = {0.35, 0.25, 0.2, 0.1, 0.1}

Below TABLE-VI shows Entropy, Average Length, Redundancy of Huffman codes and Double Huffman coding for all the samples with two types of algorithm like Huffman coding and Minimum Variance Huffman coding, here (a) shows Huffman coding and (b) shows Minimum Variance Huffman Coding.

TABLE VI: RESULTS OF DOUBLE HUFFMAN CODING USING BOTH THE APPROACHES.

Approach	First Round Coding			Second Round Coding		
	Entropy *	Average Length*	Redundancy*	Entropy *	Average Length*	Redundancy*
A	2.3375	2.4	0.0625	0.9871	1	0.012
B	2.3375	2.4	0.0625	0.9871	1	0.012
A	2.1218	2.2	0.0782	0.984	1	0.016
B	2.1218	2.2	0.0782	1	1	0
A	2.1589	2.2	0.0411	0.98	1	0.02
B	2.1589	2.2	0.0411	1	1	0

* bits/symbol

IV. RESULT ANALYSIS

We have performed Huffman coding and Minimum Variance Huffman coding with Double Huffman coding for both the approaches which gives better results. The TABLE-VII contains efficiency of both the approaches for different samples.

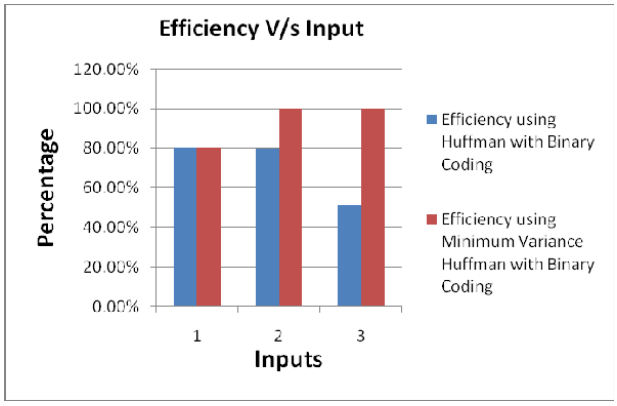

Fig 5: Efficiency V/s Input

The results are put on the bar chart, Figure 5, plots both the results of Approach A and Approach B, which clearly indicates that double coding outperform the single coding in the higher level of input. Thus it can be generalized that double Minimum Variance Huffman coding produces a better compression than the single Huffman coding and therefore it is more efficient.

TABLE - VII: EFFICIENCY OF BOTH THE APPROACH

Letters	Probability	A-1	A-2
A, B, C, D, E, F	0.30, 0.30, 0.13, 0.12, 0.10, 0.05	80.00	80.00
A, B, C, D, E	0.4, 0.2, 0.2, 0.1, 0.1	79.50	100.00
A, B, C, D, E	0.35, 0.25, 0.2, 0.1, 0.1	51.00	100.00

A-1: Efficiency using Huffman with Binary Coding
A-2: Efficiency using Minimum Variance Huffman with Binary Coding

V. CONCLUSION

The Huffman coding is a well-known lossless compression technique. We have proposed double minimum variance Huffman coding to improve the performance over Huffman method. The double minimum variance Huffman coding though increases the computational complexity and space complexity to some extent, the results show a considerable improvement in the performance. The results show a greater increase in compression for the large size of input data. We could able to achieve up to 100% efficiency for the mentioned example. However this may slightly vary with the given example but showing always an improved performance than the base method. Thus the proposed method is an excellent technique to compress text information.

REFERENCES

[1]. Mamta Sharma, "Compression Using Huffman Coding", IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.5, May 2010,pp 133-141.

[2] D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., September 1952, pp 1098–1102. Huffman's original article.

[3] R M Capocelli, R Giancarlo and I J Taneja, "Bounds on the redundancy of Huffman codes", IEEE Transactions on Information Theory, IT-32, pp 854-857, 1986.

[4] A Turpin and A Moffat, "Housekeeping for Prefix Coding", IEEE Transactions on Communications, 48, pp.622- 628, 2000.

[5] D. Huffman, "A method for the construction of minimum redundancy codes", Proc. Of the Institute of Radio Engineers, Vol 40, pp 1098-1101, 1952.

[6] Entropy: A New Definition and its Applications, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 21, No. 5, September /October 1991.